WL-TR-91-3007

An Expert System Approach to Global Fault Detection and Isolation Design

②

S. M. Allen and A. K. Caglayan

Charles River Analytics Inc.
55 Wheeler Street
Cambridge, MA 02138

January 1990

Final Report for Period June 89 – December 89

FLIGHT DYNAMICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6553

NOTICE

This technical report has been reviewed and is approved for publication.

[illegible]
BARRY S. MIGYANKO, CAPT, USAF
Flight Control Systems Development Engr

[illegible]
FRANK R. SWORTZEL, Chief
Control Systems Development Branch

[redacted]

[illegible]
JAMES E. HUNTER, Assistant Chief
Flight Control Division

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release: distribution is unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) R9004	5. MONITORING ORGANIZATION REPORT NUMBER(S) WL-TR-91-3007
6a. NAME OF PERFORMING ORGANIZATION Charles River Analytics Inc. — 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Flight Dynamics Dir. (WL/FIGL) Wright Laboratory
6c. ADDRESS (City, State, and ZIP Code) 55 Wheeler Street Cambridge, MA 02138	7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, OH 45433-6553
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Flight Dynamics Laboratory — 8b. OFFICE SYMBOL (If applicable) WL/FI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-89-C-3606
8c. ADDRESS (City, State, and ZIP Code) WL/FI WPAFB OH 45433-6553	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 65502F — PROJECT NO. 3005 — TASK NO 40 — WORK UNIT ACCESSION NO. 50

11. TITLE (Include Security Classification)

An Expert System Approach to Global Fault Detection and Isolation Design

12. PERSONAL AUTHOR(S)

S. M. Allen and A. K. Caglayan

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 89JUN26 TO 89DEC26	1991 January	50

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
01	03	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

Fault Detection and Isolation, Global FDI, Expert Systems Neural Networks, Computer-Aided Design

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This final report summarizes the results of the Phase I SBIR study entitled "An Expert System Approach to Global Fault Detection Design" supported by the U. S. Air Force under Contract #F33615-89-C-3606. The Primary objective of this research is the development and demonstration of a global Failure Detection and Identification (FDI) Design Assistant (DA) prototype based on expert system technology. A secondary objective is investigate the implementation of global FDI algorithms using the artificial intelligence techn ques of expert systems and neural networks.

A global FDI algorithm performs its detection, isolation and estimation function by assessing the global effects of a hardware fault and surface damage on the closed-loop aircraft dynamics. Since changes in the flight control law affect the signature of a hardware fault and surface damage, the design of a global FDI (continued on attached)

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Barry Miqyasko, Capt.	22b TELEPHONE (Include Area Code) (513) 255-8429 — 22c OFFICE SYMBOL WL/FIGL

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE algorithm is dependent on the flight control law performance characteristics. Ideally, the flight control law and global FDI designs should be accomplished simultaneously, especially for a new aircraft design rather than the current practice of serial iterations on the two designs. Hence, automation of the global FDI design process is desirable in order to reduce the development time, cost and risk for advanced flight control systems.

In this report we describe the automation of the global FDI design process using expert systems technology. An expert system is a computer program that mimics the problem solving skill of a human expert in a narrow domain. The development of the expert system design assistant is feasible since the domain characteristics associated with global FDI design meet the following necessary conditions. First, the design of a global FDI algorithm involves heavy use of cognitive skills. Second, there are available design experts and analysis models and programs suitable for knowledge extraction. Third, the task is neither a trivial problem nor too difficult to solve. Fourth, the task requires symbolic knowledge representation. Finally, finding a solution requires the use of heuristics and reasoning from first principles.

A secondary objective of our st [illegible] is to investigate how expert systems and neural networks can be integrated into the implementation of a global FDI system. Here we describe the implementation of the detection and isolation logic portions of the CRCA global FDI algorithm as a rule-based expert system so that symbolic knowledge representation and reasoning can be incorporated into the design. Such an approach can reduce the false alarm and misisolation rate of a given numeric algorithm by modeling the deficiencies of such an algorithm using a rule-based approach. We also present a neural net demonstration for scheduling dynamic thresholds in the CRCA global FDI.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

This final report summarizes the results of the Phase I SBIR study entitled "An Expert System Approach to Global Fault Detection and Isolation Design" supported by the U.S. Air Force under Contract No.F33615-89-C-3606. The primary objective of this research is the development and demonstration of a global Failure Detection and Identification (FDI) Design Assistant (DA) prototype based on expert system technology. A secondary objective is to investigate the implementation of global FDI algorithms using the artificial intelligence techniques of expert systems and neural networks.

Reconfigurable flight control systems (Chandler 84) make use of the "aerodynamic redundancy" in compensating for the effects of onboard hardware failure and surface damage conditions. Such fault/damage tolerant flight control systems reproduce the forces and moments acting on the airframe using the remaining available resources after the detection, isolation and estimation of the type and level of a hardware fault and surface damage. The necessary detection, isolation and estimation information is provided by a global Fault Detection and Isolation (FDI) system which monitors the flight control and surface position sensors (Caglayan et al. 87).

A global FDI algorithm performs its detection, isolation and estimation function by assessing the global effects of a hardware fault and surface damage on the closed-loop aircraft dynamics. Since changes in the flight control law affect the signature of a hardware fault and surface damage, the design of a global FDI algorithm is dependent on the flight control law performance characteristics. Ideally, the flight control law and global FDI designs should be accomplished simultaneously, especially for a new aircraft design rather than the current practice of serial iterations on the two designs. Hence, automation of the global FDI design process is desirable in order to reduce the development time, cost and risk for advanced flight control systems.

In this report we describe the automation of the global FDI design process using expert systems technology. An expert system is a computer program that mimics the problem solving skill of a human expert in a narrow domain. The development of the expert system design assistant is feasible since the domain characteristics associated with global FDI design meet the following necessary conditions. First, the design of a global FDI algorithm involves heavy use of cognitive skills. Second, there are available design experts and analysis models and programs suitable for knowledge extraction. Third, the task is neither a trivial problem nor too difficult to solve. Fourth, the task requires symbolic knowledge representation. Finally, finding a solution requires the use of heuristics and reasoning from first principles.

The FDI/DA is an expert system based computer aided design environment that assists designers in the global FDI design process. The system incorporates a data base of failure detection and isolation details and procedures, and is capable of suggesting design reiteration procedures to the user. Currently, the FDI/DA expert system operates on a MicroVAX II workstation and interfaces with the global FDI software for the Control Reconfigurable Combat Aircraft (CRCA) (Weinstein et al., 87).

A secondary objective of our study is to investigate how expert systems and neural networks can be integrated into the implementation of a global FDI system. Here we describe the implementation of the detection and isolation logic portions of the CRCA global FDI algorithm as a rule-based expert system so that symbolic knowledge representation and reasoning can be incorporated into the design. Such an approach can reduce the false alarm and misisolation rate of a given numeric algorithm by modelling the deficiencies of such an algorithm using a rule-based approach. We also present a neural net demonstration for scheduling dynamic thresholds in the CRCA global FDI.

1.1. Global FDI Overview

Here, we give an overview of the global FDI system used in the second generation reconfiguration strategy for aircraft flight control systems subjected to actuator failure/surface damage developed under the SRFCS program (Chandler 84). In this program, three competing reconfiguration strategies have been developed during the Phase I effort (Caglayan et al. 87) (Lear Siegler 87), and (Scientific Systems 86). The Failure Detection and Isolation (FDI) system in (Caglayan et al. 87) and the Reconfigurable Control Law (RCL) in (Lear Siegler 87) have been integrated into a unified reconfiguration strategy, and the unified reconfiguration strategy design has been transitioned onto the Control Reconfigurable Combat Aircraft (CRCA) (Weinstein et al. 87) under the Phase II effort. The preliminary performance results for the unified reconfiguration strategy have been presented in (Caglayan, Allen, and Wehmueller 88).

The reconfiguration strategy, as shown in Figure 1.1-1, consists of a Reconfigurable Control Law (RCL) composed of a robust flight control system tolerant of low level surface damage which is reconfigured after impairment to recover performance and minimize transients; a hierarchical Failure Detection Isolation and Estimation (FDIE) system identifying actuator failures and moderate-to-severe surface damage; and a Reconfiguration Logic coordinating the information transfer between the RCL and FDIE systems. In the developed reconfiguration strategy, the robust flight control system provides time for the FDIE system to detect and isolate the failures. Moderate-to-high surface damage effects are compensated based on the control effectiveness parameter estimates provided by the FDIE system. Low level surface damage effects are handled by the robust flight control system with the effectiveness estimates stored for use in case of subsequent failures.

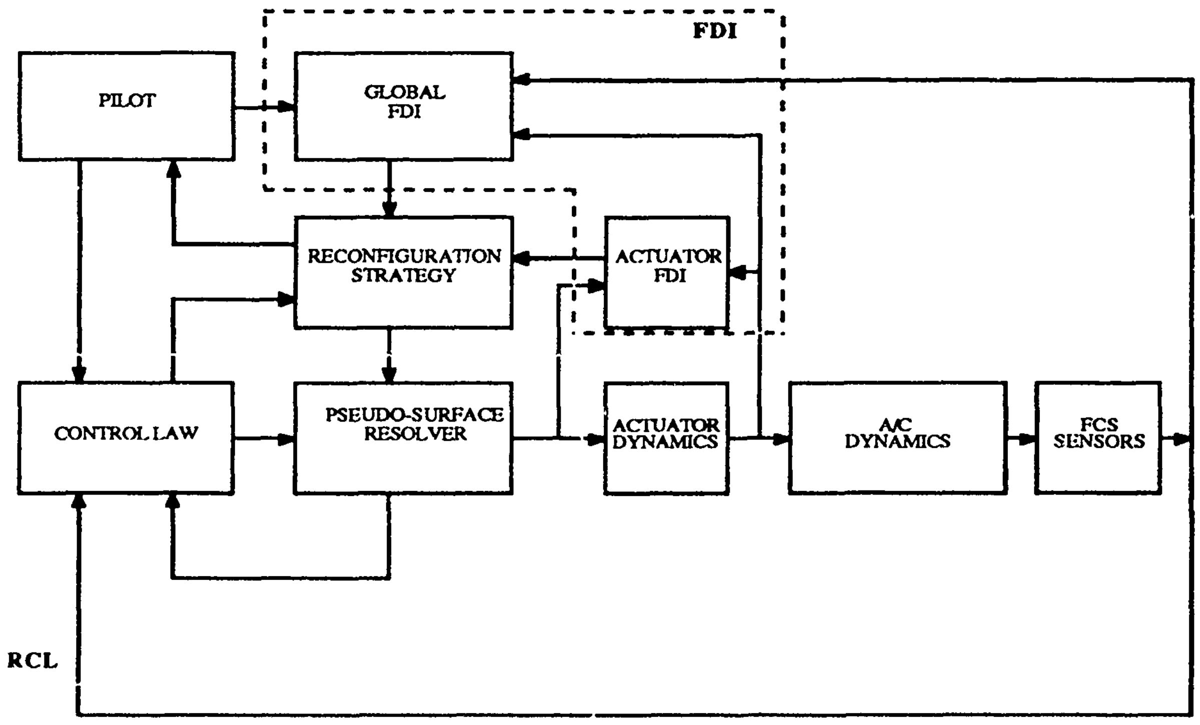

Figure 1.1-1: Reconfiguration Strategy Block Diagram

The FDIE system consists of a local Actuator Failure Detection (AFD) system which detects stuck, runaway, and floating actuator failures based on local information; and, a global Surface Damage Detection and Isolation (SDDI) system which detects partial surface loss failures and provides an estimate for the surface control effectiveness parameters after the impairment based on global information. The global SDDI algorithm accounts for expected modelling errors in the aircraft dynamics description and uses a modified multiple hypothesis test to declare a full isolation only when a certain distinguishability criterion is met.

The CRCA FDI system uses local information for actuator failure detection and global information for surface damage detection. The local information used for actuator failure detection consists of actuator dynamics, flight control system commands into the actuator, and the actuator output measurement. On the other hand, the global information used for surface damage detection consists of aircraft force and moments equations providing a dynamic relationship between the surface position sensors, and flight control sensor measurements. The global information can help detect and isolate actuation failures that are undetectable by local information analysis.

The global FDI in the CRCA reconfiguration strategy algorithm, shown in Figure 1.1-2, consists of:

* a *no-fail filter* based on aircraft equations of motion and expected modelling uncertainty under no surface damage conditions computing the measurement residuals for flight control sensors;

* a bank of first order filters (called *detectors*) compensating the no-fail filter residuals based on the computed partial surface loss estimates and on the expected modelling errors;

* a bank of *likelihood ratio* computers computing the likelihood of each postulated surface damage hypothesis using the compensated residuals;

* a *modified multiple hypothesis test* for making surface damage detection, partial isolation to a subset of surfaces, and full isolation decisions.

The no-fail filter is an extended Kalman filter based on the linearized aerodynamic force and moment equations with nonlinear inertial terms and on the dynamic modelling uncertainty during maneuvers. The inputs into the no-fail filter are the surface position measurements, and the FCS sensor outputs comprised of body mounted accelerometers and rate gyros, indicated airspeed (IAS), inertial measurement unit (IMU) attitude, angle of attack, and sideslip measurements. The output of the no-fail filter is the FCS measurement residuals vector which is the difference between the FCS sensor outputs and their predictions computed by the no-fail filter.

Associated with each surface, there is a first order filter estimating the percentage gain reduction in control effectiveness for that surface assuming that the surface in question has been damaged at a random instant with an unknown damage level. The input to each detector is the no-fail filter FCS measurement residual vector. The detectors compute the covariance for this measurement residual vector to account for the modelling effects on the no-fail filter residuals during maneuvers by using a state and input dependent process noise (Caglayan, Allen and Rahnamai, June 89).

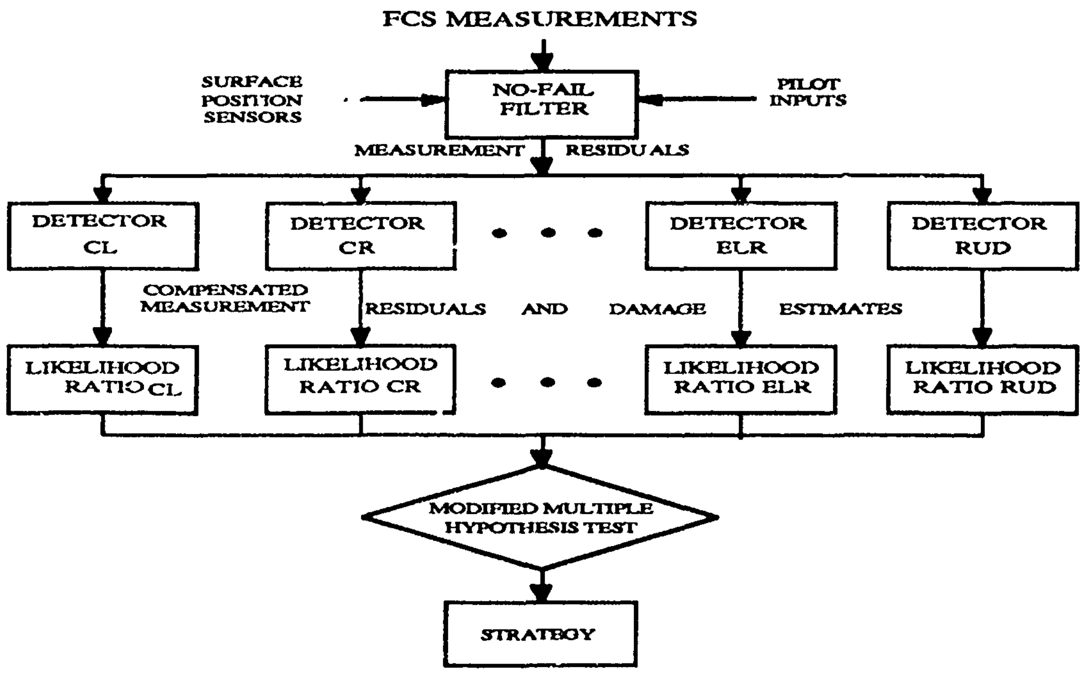

Figure 1.1-2: Global FDI System

The output of each detector is the estimate for the control effectiveness loss and the associated covariance reflecting the uncertainty about the damage estimate. Each detector also outputs a compensated FCS sensor measurement residual sequence such that the effects of the hypothesized surface damage are removed from the no-fail filter residuals.

For each detector, a likelihood ratio is computed using the compensated residual sequence from that detector over a fixed moving window. Each likelihood ratio is a weighted sum of the compensated measurement residuals adjusted by detection thresholds.

In the modified multiple hypothesis test, first, a detection test is performed which yields a set of candidates for damaged surfaces. Next, a full isolation test is performed which declares complete isolation among the damaged surface candidates only when a specific distinguishability criterion is satisfied. The modified multiple hypothesis test therefore has two sets of thresholds: I) detection thresholds for the detection with partial isolation test; II) isolation thresholds for the full isolation tests. These thresholds are functions of flight conditions.

1.2. Expert Systems Overview

The recent success of expert systems technology (Stefik et al. 82), a subfield of Artificial Intelligence, in diagnosis, monitoring, prediction, planning tracking and design problems in certain application domains has initiated similar efforts in other areas. These early successful expert system applications include SOPHIE in computer assisted instruction (Brown, et al.), MYCIN in medical diagnosis (Shortcliffe 76), PROSPECTOR in oil exploration (Duda et al. 78), and DENDRAL in biology (Buchanan and Feigenbaum 78). Recent military applications of expert systems include ADEPT for battlefield situation assessment analysis (Taylor et al. 84), AIRPLAN for assisting air operations officers with the launch and recovery for aircraft on a carrier (Masui et al. 83), EXPERT NAVIGATOR for monitoring aircraft navigation sensors (Pisano and Jones 84) aircraft and B1B diagnostic expert system (Davis 88).

An expert system is a computer program that can perform a task normally requiring the reasoning ability of a human expert. Expert systems are highly specialized according to their application domains. Although any program solving a particular problem may be considered to exhibit expert behavior, expert systems are differentiated from other programs according to the manner in which the domain specific knowledge is structured, represented and processed to produce solutions. In particular, expert system programs partition their knowledge into the following three blocks: Data Base, Rule Base, and Inference Engine. Expert systems utilize symbolic and numeric reasoning in applying the rules in the Rule Base to the facts in the Data Base to reach conclusions according to the construct of reasoning specified by the Inference Engine.

1.2.1. Knowledge Representation

There are two basic types of knowledge that can be incorporated into expert systems: declarative knowledge and procedural knowledge. The kind of knowledge describing the relationships among objects is called declarative knowledge. The kind of knowledge prescribing the sequences of actions that can be applied to this declarative knowledge is called procedural knowledge. In expert systems, procedural knowledge is represented by production rules whereas declarative knowledge is represented by frames, semantic networks in addition to production rules.

Rules are expressed as IF-THEN statements. When the IF portion of a rule is satisfied by the facts, the rule is fired by executing the statements specified by the THEN portion. Typically, the production rules deal with uncertainty through the use of certainty factors, probability or fuzzy logic. Semantic nets are network representations of declarative knowledge. A semantic net consists of a collection of nodes - representing arbitrary objects - connected by arcs describing the relations between nodes. One of the most important characteristics of a semantic net is the capability of building inheritance hierarchies. Using arcs representing relations such as IS-A and HAS-PART, objects in the net can inherit properties from other objects higher up in the net. A frame is a knowledge representation about a prototypical instance (Fikes and Kohler 85). Frames are organized as semantic nets where the topmost nodes represent general concepts whereas the lower nodes represent more specific instances of these concepts. In a frame based system, the concept at each node is defined by its attributes (slots) and attribute values. Each slot can also contain procedures which are executed when the values of the attribute change.

While expert systems have been traditionally built using collections of rules based on empirical associations, interest has grown recently in knowledge-based expert systems which perform reasoning from representations of structure and function knowledge. For instance, an expert system for digital electronic systems troubleshooting is developed by using a structural and behavioral description of digital circuits (Davis 84). The objective of this approach to expert system implementation is to reason from first principles about the domain rather than from empirical associations. One of the key ideas in this approach is to use multiple representations of the digital circuit (both functional and physical structure) in troubleshooting applications. The approach is also similar to the multiple levels of abstraction in modelling of mental strategies for fault diagnosis problems (Rasmussen 85).

Qualitative process theory (Forbus 88) is another approach allowing the representation of causal behavior based on a qualitative representation of numerical knowledge using predicate calculus. QP theory is a first order predicate calculus defined on objects parameterized by a quality consisting of two parts: an amount and a derivative, each represented by a sign and magnitude. In Qualitative Process theory, physical systems are described in terms of a collection of objects, their properties, and the relationships among them within the framework of a first order predicate calculus. Hierarchical knowledge representation at several levels of abstraction is also another approach used in modelling human problem solving strategies for complex systems (Rasmussen 85). This hierarchy is two dimensional. The first is the functional layers of abstraction for the physical system: functional purpose, abstract function, generalized function, physical function, and physical form. The second is the structural layers of abstraction for the physical system: system, subsystem, module, submodule, component.

1.2.2. Inference Strategies

The inference control strategy is the process of directing the symbolic search associated with the underlying type of knowledge represented in an expert system: antecedents of IF-THEN rules, nodes of a semantic net, or collection of frames. In practical expert system applications, the blind search is an unacceptable approach due to the associated combinatorial explosion. Search techniques can be basically grouped into three: breadth-first, depth-first and heuristic. The breadth-first search exhausts all nodes at a given level before going to the next level. In contrast, the depth-first exhausts all nodes in a given branch before backtracking to another branch at a given level. Heuristic search incorporates general and domain-specific rules of thumb to constrain a search.

Expert systems employ basically two types of reasoning strategies based on the search techniques above: forward chaining and backward chaining. In forward chaining, starting from what is initially known, a chain of inferences is made until a solution is reached or determined to be unattainable. For instance, in rule based systems, the inference engine matches the left-hand side of rules against the known facts, and executes the right-hand side of the rule that is activated. In contrast, backward-chaining systems start with a goal and searches for evidence to support that goal. Pure forward chaining is appropriate when there are multiple goal states and a single initial state whereas backward chaining is more appropriate when there is a single goal state and multiple initial facts. Many expert systems utilize both forward and backward chaining.

1.3. Neural Networks Overview

Neural networks (Anderson and Rosenfeld, 88) represent nonalgorithmic class of information processing for using massively parallel distributed processing architectures. Stimulated by the efforts directed at understanding the interconnection of neurons in the human brain allowing the storage, retrieval, and processing of complex data, research over the last 25 years in artificial neural systems has produced solutions to complex problems in visual pattern recognition, combinatorial search, and adaptive signal processing.

A generic artificial neural net structure is a network of processing elements (neurons) connected with each other through interconnects (information links). Each processing element can have multiple inputs and only one output. The input/output relationship is described by a first-order differential equation. Specifically, a weighted sum of the nonlinear transformations of the multiple inputs along with a nonlinear transformation of the current neuron's state are the driving functions of this first-order differential equation. The weighting coefficients also satisfy a first-order differential equation driven by a nonlinear transformation of multiple inputs and the weighting coefficients associated with the input links. These two first-order differential equations (also called the update and learning rules) and a specific network topology provide a complete neural network specification.

Artificial neural networks produce a nearest-neighbor classifier. Since the weighting coefficients change in an unpredictable manner, the global stability of the neural network description is an important consideration. The strongest theoretical result to date is due to (Cohen and Grossberg, 83) who have shown that the neural net converges to one of the finite set of equilibrium points corresponding to local minima of the energy function under certain restrictive conditions (e.g., symmetric, positive weighting coefficients). Another advantage of neural nets is that the convergence to the answer is independent of the number of local minima in the energy function, thus comparing favorably to other general search techniques. Although the global stability and convergence results have not been extended to the case for nonsymmetric weighting coefficients, several successful heuristic applications with nonsymmetric weighting coefficients have been reported (Grossberg, 82), (Hecht-Nielsen, 86).

Conceptually, artificial neural networks are applicable to failure detection problems for learning the spatiotemporal patterns associated with failure signatures. Here, the spatiotemporal pattern would be the measurement residual trajectory as a function of time in the measurement vector space. The relationship between artificial neural network approach to spatiotemporal pattern recognition and conventional signal processing structures (i.e., finite impulse filters, correlation detectors, etc.) has been discussed by (Myers, 86). Used in this context for FDI application, neural networks would be the nonalgorithmic counterpart of spatiotemporal filters. The direct application would involve the training of a neural network representation of a structure with a multitude of failure scenarios and using the neural net as a classifier. Neural networks can also be applied to model empirical relationships for covariance expressions, likelihood ratio thresholds under modelling errors.

The recent interest in artificial neural networks is due to the availability of fast, relatively inexpensive computers made possible by advances in VLSI design for realizing neural network structures. Given that the neurons in the human brain process information in milliseconds while outperforming current serial supercomputers with a processing rate in nanoseconds, there is considerable interest in the new generation neurocomputers and computing environments.

1.4. Outline of Report

In chapter 3, we describe the development and implementation of the design assistant prototype. We present the formulation of the FDI/DA prototype in Chapter 2. Expert system and neural network implementations of global FDI algorithms are described in chapter 4. In Chapter 5 we describe the requirements of a full scale FDI/DA research prototype.

2. FORMULATION OF PROBLEM

2.1. Determination of Problem Scope

Global FDI design is an iterative and time consuming process which requires the knowledge and expertise of an expert FDI designer. There are three main steps in the global FDI design process, which is illustrated in Figure 2.1-1. First, the FDI design expert analyzes aircraft and FDI time history data to produce an assessment of the FDI performance for the simulated damage scenarios. Second, if the designer is not satisfied with the FDI performance he uses his domain expertise to determine FDI design parameter modifications which will improve FDI performance. The designer also determines simulation scenarios which will test that the FDI design parameter modifications improve FDI performance. Third, the designer runs the simulation scenarios, which results in new time history data, and the process repeats until the desired FDI performance is reached. The third step also includes running a matrix of simulation scenarios in order to ensure that the design modifications do not adversely affect FDI performance.

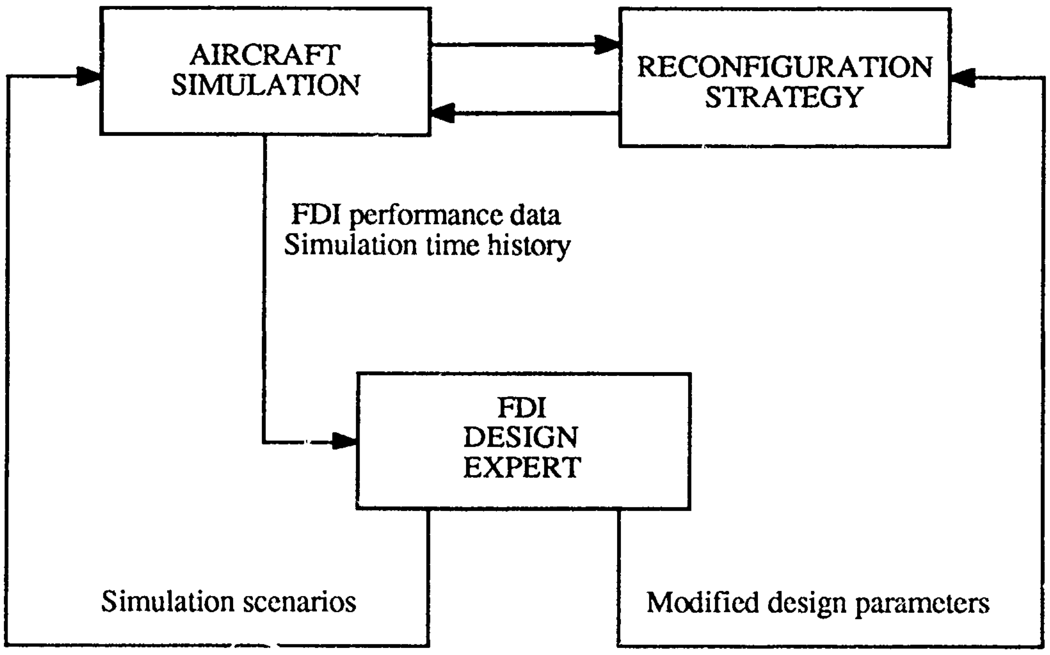

Figure 2.1-1: FDI Design Process

Increasing the efficiency of the global FDI design process requires achieving two main development goals. The first goal is to decrease the amount of time required for an expert FDI designer (someone with expertise in both FDI theory and aero application knowledge) to design the global FDI. This goal can be achieved by automating FDI performance analysis and the running of simulation scenarios. The second goal is to enable a designer with aero application knowledge expertise but minimal FDI expertise to design the global FDI. In order to achieve this goal, the inexperienced FDI designer will need assistance in determining design parameter modifications and selecting simulation scenarios.

2.2. Determination of Development Environment

In addition to the conventional AI programming languages (i.e., various flavors of LISP and PROLOG), general purpose knowledge engineering languages such as OPS5, CLIPS, and M.1 can be used to build customized expert systems. Several commercial integrated expert system development tools are also available for building expert system applications. General purpose commercial expert system shells for building expert systems can be grouped into three classes: rule-based, frame-based and hybrid. In rule-based expert system shells, the domain knowledge is captured using if-then decision rules. These rules capture informal heuristics about a given domain. In frame-based expert system shells, domain knowledge is captured as data structures that relate the characteristics of objects in a hierarchy of classes. The organization of frames into taxonomic hierarchies provide an efficient method of knowledge representation through inheritance. Hybrid expert system shells provide a combination of rules, frames and inheritance networks using object oriented representation with multiple inheritance. Hence, hybrid shells allow the partitioning of domain knowledge for representation.

The expert system shell CLIPS (Giarratano 88) is chosen as the development environment for the FDI Design Assistant expert system. CLIPS - C Language Integrated Production System - is a tool for the development of rule based Expert Systems. CLIPS provides a powerful rule syntax and an inference engine based on the Rete match algorithm (Forgy 82). We have selected CLIPS for the FDI/DA implementation since it is written in C, embeddable to other programs written in different languages (C, Fortran, Ada), and portable across hardware platforms. Although CLIPS is a pure rule-based expert system de\ elopment tool (e.g. it does not support inheritance), the production system models have been advocated as computational models for human cognition (Anderson 83), thus suitable for representing the FDI design expert's knowledge in terms of production rules.

2.3. Performance of Knowledge Engineering

Knowledge engineering is the process of transferring a domain expert's knowledge from the expert to a computer program or expert system. The knowledge engineering process for the FDI/DA is illustrated in Figure 2.3-1. The knowledge engineer is responsible for converting an experts knowledge into the facts and rules that make up the expert system. This is accomplished by interviewing the expert, observing and formalizing the expert's reasoning during the design process.

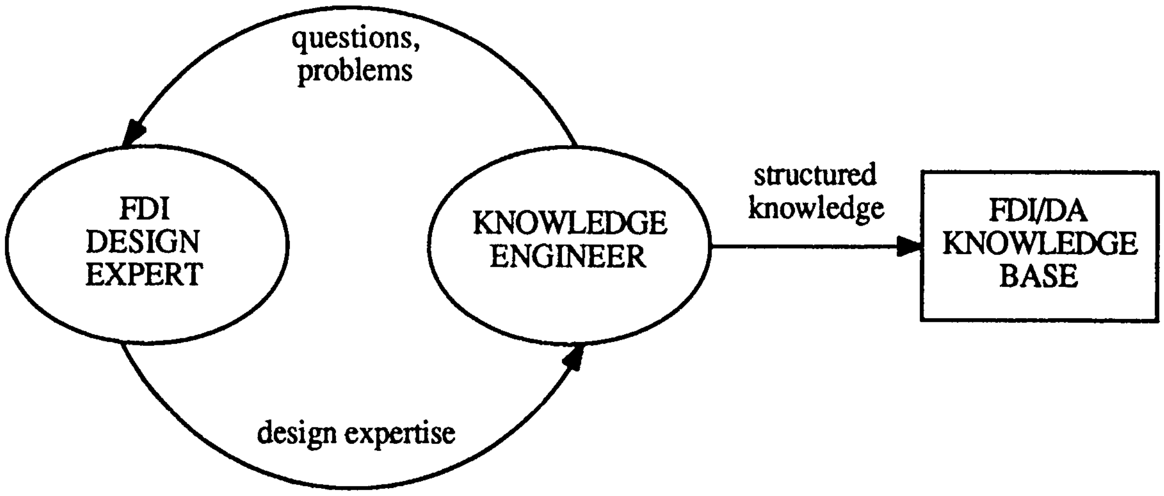

Figure 2.3-1: FDI/DA Knowledge Engineering Process

3. FDI/DA PROTOTYPE IMPLEMENTATION

The FDI/DA prototype is implemented on a Microvax II workstation running under MicroVMS V4.5. The prototype interfaces with the non-real time CRCA simulation, version 7. The control and data flow structures of the FDI/DA is shown in Figure 3-1. As seen in the figure the FDI/DA consists of three main sections: the fact list, the rule base, and the inference engine. The fact list consists of facts about FDI performance, which are input from the CRCA simulation, facts about the current design process, and facts concerning user inputs. The rule base has rules for processing user inputs, rules for FDI performance analysis, and rules for design iteration. The inference engine constantly looks for matches between the facts in the fact list and and the left hand sides of the rules in the rule base. When a match occurs, the matched rule is put on an agenda. The inference engine then selects a rule from the agenda and executes, or fires, the rule. The firing of the rule can place additional facts on the fact list and can cause other rules to fire.

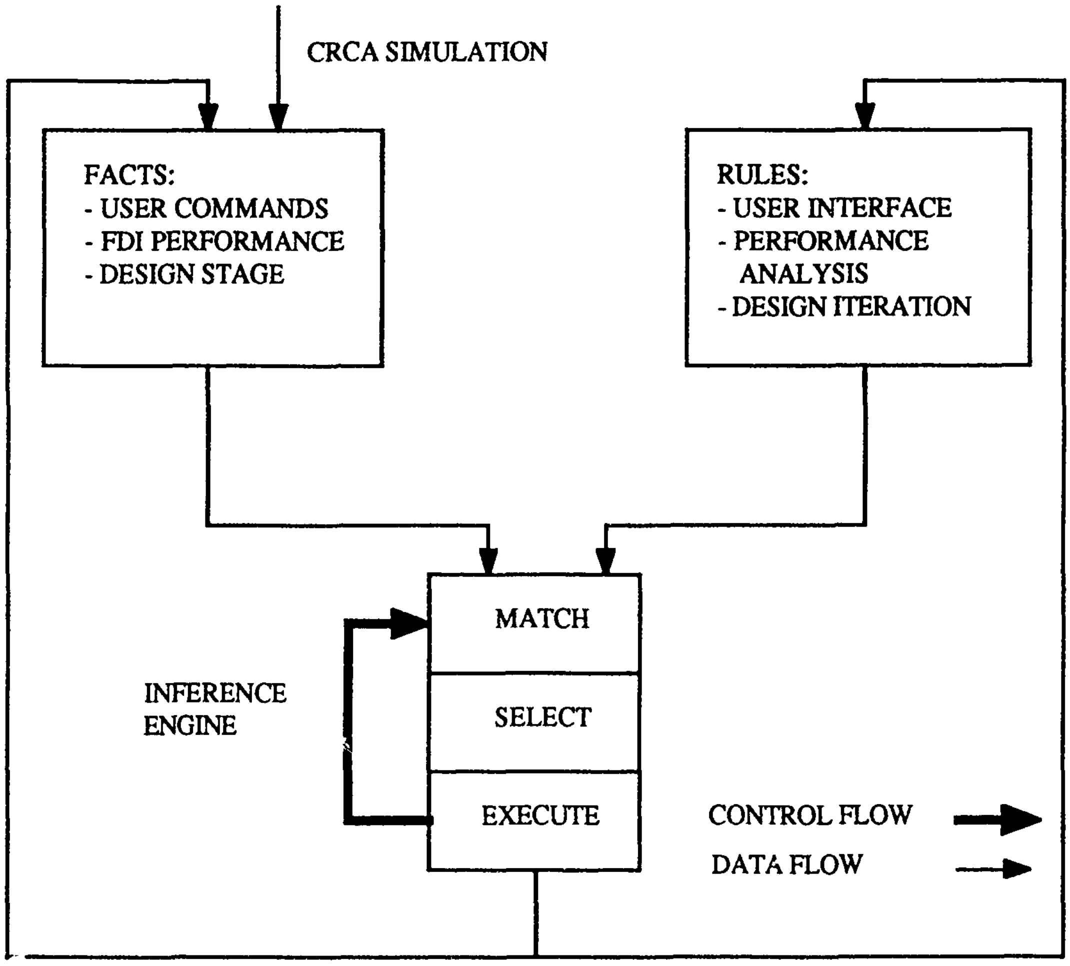

Figure 3-1: FDI Control and Data Flow

3.1. FDI/DA Knowledge Base

3.1.1. FDI/DA Fact List

As mentioned earlier, the fact list consists of facts concerning FDI performance, the current design process, and user inputs. An initial set of facts is placed in the fact list when the FDI/DA is started. Additional facts are asserted into the list when certain rules fire. The firing of a rule can cause facts to be removed from the fact list.

Figure 3.1.1-1 shows the facts that initialize the fact list when the FDI/DA is started. The CLIPS syntax for specifying initial facts is the *deffacts* statement. The first fact asserted into the fact list, prompt-user, is a user command fact and it matches a pattern in the left-hand side of the prompt rule. When the prompt rule fires it prompts the user to enter an FDI/DA command. The flight-condition and threshold-abb facts are used in design rules for determining simulation script file names. The facts included in the group of facts named current-design-state describe what is known about the current FDI design. As shown in this group of facts, there is initially minimal information about the current design.

```
(deffacts initial-facts
 (prompt-user)
)

(deffacts abbreviations
  (flight-condition (name stol) (abbreviation s))
  (flight-condition (name exit) (abbreviation x))
  (flight-condition (name entry) (abbreviation e))
  (flight-condition (name tfta) (abbreviation t))
  (threshold-abb (name static-detection) (abbreviation sd))
  (threshold-abb (name static-isolation) (abbreviation si))
  (threshold-abb (name dynamic-detection) (abbreviation dd))
  (threshold-abb (name dynamic-isolation) (abbreviation di))
)

(deffacts current-design-state
 (design-flight-condition unknown)
 (safety-margin (name threshold) (value 20))
 (script-file)
 (reason-why one
"The 'why' command explains why the last conclusion printed to the screen
was reached. Type 'why' at the FDI/DA command line prompt, but not in
response to an input request.")
)
```

Figure 3.1.1-1: Initial FDI/DA Fact List

3.1.2. FDI/DA Rule Base

The FDI/DA knowledge base is a representation of FDI design expert's knowledge as a set of production rules and facts. The rule base has rules for processing user inputs, analyzing FDI performance, and iterating the current design. When *patterns* on the left hand side of a rule match facts in the fact list, the rule fires.

Figure 3.1.2-1 shows three of the rules used for processing user inputs. In CLIPS, rules are defined using the *defrule* statement. The rule named prompt will fire when the fact prompt-user is in the fact list. The salience statement helps the inference engine determine which rule to fire if more than one rule is on the agenda. The higher the salience the higher priority a rule has. When this rule fires the FDI/DA command line prompt **DA>>** is printed on screen and the system waits for the user to input a command. After a command has been entered, it is converted from a string into a fact using the str-explode function, it is then asserted into the fact list using the (assert (command $?command)) statement. The next rule, unknown-command, is used to alert the user when an invalid command has been entered. The reset-prompt rule reasserts the prompt-user fact into the fact list so that the prompt rule will fire and prompt the user for the next command. The salience on the reset-prompt rule is set to -100 to ensure that the actions resulting from the last command have finished executing before the user is prompted for another command.

```
(defrule prompt
  (declare (salience -25))
 ?prompt <-  (prompt-user)
=>
 (retract ?prompt)
 (printout t crlf)
 (printout t "DA>> ")
 (bind ?string (readline))
 (bind $?command (str-explode ?string))
 (assert (command $?command))
)

(defrule unknown-command
  (declare (salience -50))
  (command ?command&~help&~diag&~gt&~why&~facts&~rules
           &~agenda&~watch&~unwatch [illegible] ~quit&~fc&~ed
           &~dir&~load&~trans&~cfa&~[illegible]
          &~time $?command-2)
=>
 (printout t "*** "  ?command " "
           $?command-2 " is not a valid command! ***" crlf)
)

(defrule reset-prompt
  (declare (salience -100))
 ?command <- (command $?)
=>
 (retract ?command)
 (assert (prompt-user))
)
```

Figure 3.1.2-1: FDI/DA User Interface Rules

Examples of FDI/DA rules for analyzing FDI performance are shown Figure 3.1.2-2. The first rule determines whether or not there was an FDI false alarm during the simulation run. The fact check for FDI anomalies is asserted into the fact list when the user inputs the command **cfa** at the command line prompt. The run-fact facts are asserted into the fact list after the FDI time history data generated during a CRCA simulation run has been translated into a CLIPS readable form.

The first rule reads as follows: If there was not a failure and damage was detected, then there was a false alarm. The second rule determines whether or not canard damage was isclated correctly.

```
(defrule check-for-false-alarms
  (check for FDI anomalies)
  (run-fact there was-not a-failure)
  (run-fact damage was detected)
=>
  (assert (there was a-false-alarm))
)

(defrule check-for-correct-canard-isolation
  (check for FDI anomalies)
  (run-fact there was a-failure)
  (run-fact damage was detected)
  (run-fact damage was isolated)
  (run-fact failure ?failure&cl|cr $?)
  (run-fact isolated-damage ?iso-dam&cl|cr $?)
  (test (eq ?failure ?iso-dam))
=>
  (assert (there was-not an-incorrect-canard-isolation)))
```

Figure 3.1.2-2: FDI Performance Analysis Rules

The FDI/DA rule base also contains a fundamental set of rules for determining static detection thresholds. These design iteration rules assist the user in setting static detection thresholds for four flight conditions: STOL, TF/TA, ACM Entry and ACM Exit. Figure 3.1.2-3 shows two of the rules used in the design iteration process. When the user enters the command to design threshold at the command line prompt, the rule prompt-user-for-type-of-threshold- to-design fires and the user is asked which type of threshold they want to design. The user then picks an option from the menu that appears. After an option has been entered the FDI/DA will tell the user which simulation script file to run in order to generate the FDI time history data needed to design the thresholds requested. The current version of the FDI/DA will only design static thresholds.

```
(defrule prompt-user-for-type-of-threshold-to-design
  ?get <- (get threshold-type)
  (design-flight-condition ?flight-condition&~unknown)
=>
  (retract ?get)
  (bind ?fc (upcase ?flight-condition))
  (printout t crlf crlf)
  (printout t "**** Design which threshold for flight condition " ?fc "? ****" crlf)
  (printout t "*        *" crlf)
  (printout t "* OPTIONS    DESCRIPTION                    *" crlf)
  (printout t "*   sd     static detection             *" crlf)
  (printout t "*   si     static isolation             *" crlf)
  (printout t "*   dd     dynamic detection             *" crlf)
  (printout t "*   di     dynamic isolation             *" crlf)
  (printout t "*                                  *" crlf)
    (printout  t  "**************************************************************"  crlf)
  (printout t crlf "Enter an option: ")
  (assert (threshold-option =(read)))
)

(defrule tell-user-to-run-a-script-file
  ?run <- (run script-file ?filename)
=>
  (retract ?run)
  (printout t "Exit to TAE to run the following script file: " ?filename crlf)
  (printout t " At the TAE prompt, type the following: ena-scr " ?filename crlf)
)

(defrule generate-static-detection-thresholds
  (generate static-detection thresholds)
  (run-fact max-lrd surface-type ?surface ?max-lrd)
  (safety-margin (name threshold) (value ?reliability))
=>
  (bind ?threshold (+ ?max-lrd (* ?max-lrd (* ?reliability 0.01))))
  (assert (threshold (surface ?surface) (name static-detection)
           (value ?threshold)))
)
```

Figure 3.1.2-3: FDI/DA Design Iteration Rules

3.2. FDI/DA Prototype Demonstration

In this section of the report we present an interactive session with the FDI/DA for designing static detection thresholds for the ACM Entry flight condition. The output that the FDI/DA presents to the user will be shown in this font and user inputs will be in **bold** print.

After the FDI/DA has been started and the welcome message has appeared, the user can input a command at the command line prompt, DA>>. Typing help produces a list of valid commands.

```
         Welcome to the FDI Design Assistant
          type 'help' for a list of commands

DA>> help

*********** FDI DESIGN ASSISTANT COMMANDS ********
* trans          translate CRCA time history to facts        *
* load           load translated facts into fact base        *
* cfa            check for FDI anomalies                     *
* dt             design thresholds                           *
* gt             generate new threshold values               *
* tae            exit to TAE to run script files             *
* why            explain why a conclusion was reached        *
******************* CLIPS COMMANDS *****************
* facts          display facts in the fact list              *
* rules          display names of rules in data base         *
* agenda         show activated rules                        *
* watch          watch facts, activations, or all            *
* unwatch        deactivates watch command                   *
* quit           exit to CLIPS                               *
****************** SYSTEM COMMANDS *****************
* ed 'filename'  edit 'filename', e.g. 'ed fdifil.dat'       *
* dir 'files'    directory of 'files'                        *
****************************************************
```

In this example the user wants to design thresholds so the command dt will be entered. At this point the FDI/DA does not know which flight condition the design is for so the user is prompted for the design flight condition. After a valid flight condition is entered, the FDI/DA presents the user with a menu from which to chose the type of threshold to design. Since the user wants to design static detection thresholds the **sd** option is chosen. Now the FDI/DA has enough information to suggest a course of action: Exit to TAE to run the following script file: "ESDT_DSG.SCR". The user can inquire into the reasoning behind the suggested course of action by using the **why** command.

```
DA>> dt
The current design flight condition is UNKNOWN.
Enter the new design flight condition (entry, exit, stol, tfta): entry
 The current design flight condition is ENTRY.

**** Design which threshold for flight condition ENTRY? *****
*                                                           *
* OPTIONS   DESCRIPTION                                     *
*  sd       static detection                                *
*  si       static isolation                                *
*  dd       dynamic detection                               *
*  di       dynamic isolation                               *
*************************************************************
Enter an option: sd
Exit to TAE to run the following script file: "ESDT_DSG.SCR"
 At the TAE prompt, type the following: ena-scr "ESDT_DSG.SCR"

DA>> why
"In order to design a surface's static detection thresholds a run
with maximum maneuvers, maximum gusts, and no damage needs to be made.
The xSDT_DSG.SCR file will automatically make such run for the current
design flight condition [x]. A surface's static detection thresholds should be
set to some percentage greater than the surface's maximum likelihood ratio
difference [LRD] generated during this run."
```

To run the scriptfile the user exits to TAE using the **tae** command and enables the scriptfile at the TAE prompt. After the scriptfile has finished running the FDI/DA automatically restarts and suggests values for the static detection thresholds at the ACM Entry flight condition. Again, the user can use the **why** command for an explanation.

```
DA>> tae
TAE>ena-scr esdt_dsg.scr


Set the CANARD static-detection threshold for ENTRY to   1.00
Set the TRAILING-EDGE static-detection threshold for ENTRY to   1.43
Set the RUDDER static-detection threshold for ENTRY to   1.00

DA>> why
"These values are a percentage greater than the maximum likelihood
ratio, for these surface types, generated during the design run.
The percentage increase is the value in the fact 'safety-margin'
and can be modified if necessary for desired performance."
```

4. ARTIFICIAL INTELLIGENCE IMPLEMENTATION OF FDI ALGORITHMS

Artificial intelligence technologies can be used to enhance the performance of the global FDI strategy. We implemented global FDI algorithms using expert system production rules and neural networks to demonstrate the potential these technologies have for improving global FDI performance.

4.1. Expert System Implementation of Global FDI Strategy

In this section, we discuss the implementation of the global FDI as an expert system. We demonstrate the feasibility of expert system implementation of the global FDI algorithms by writing the isolation logic of the CRCA global FDI algorithm as a set of production rules and integrating it into the existing FDI code.

4.1.1. Expert System Implementation of Global FDI Algorithms

Current global FDI systems can be described using the block diagram shown in Figure 4.1.1-1. In these systems, the monitored signals drive a predictor which is either a closed loop or an open loop *numerical* simulation based on the assumption of current postulated faults in the monitored signals. The predictor provides a discrepancy called the residual between the monitored signal and its expected value.

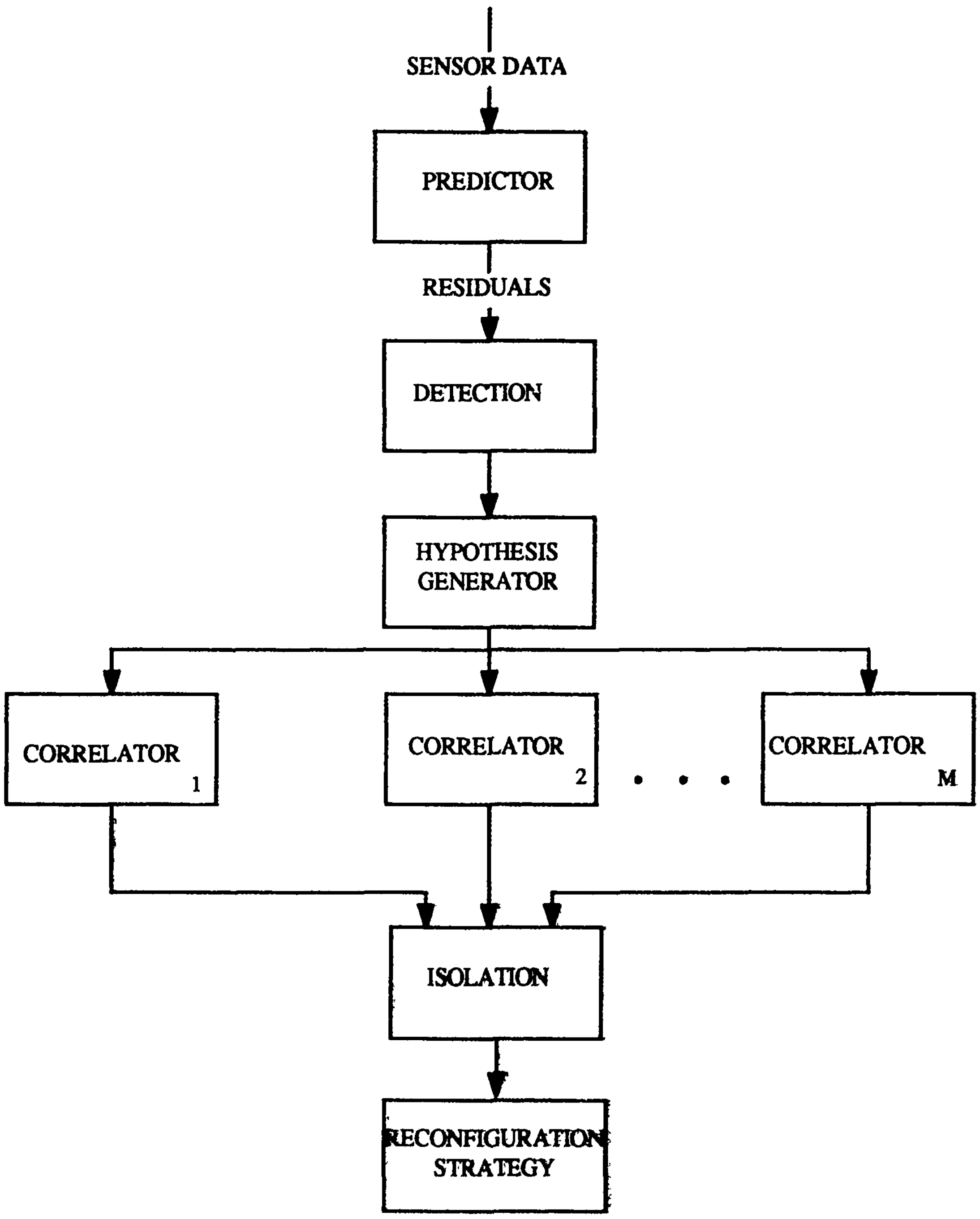

Figure 4.1.1-1: Generic FDI System Structure

If the detection test indicates the presence of a fault, then the residuals are processed through several hypothesis conditioned correlators to generate a set of compensated residuals. Then, a multiple hypothesis test is employed to isolate the fault. This is followed by a module performing reconfiguration of the predictor and correlator blocks.

What are the common problems encountered in these fault/event monitoring applications? Most of the serious problems are due to false alarms, incorrect isolations, and missed detections. The main reason for these shortcomings are due to the inaccuracy of the *numerical* model used for the system in designing either the predictor or detection or isolation algorithm.

In a conventional approach, how does a designer try to eliminate these problems? He has two options: First, he can use a better numerical model for the predictor. For instance, the model can be modified to incorporate the effects of attack changes on the aircraft stability and control derivatives. A second choice is to accept the inaccuracy of the predictor model and to account for the inaccuracy of the model. For example, the test thresholds can be dynamically changed to compensate for the effect of modelling errors on the residuals during maneuvers.

In short, the designer either has to increase the complexity of his predictor or detection and isolation algorithm by using a more elaborate *numerical* model in conventional fault monitoring and diagnosis applications. So where can expert systems be of help in theses applications? We believe that there are several areas in which expert systems technology can be of help. There are:

- *Symbolic knowledge representation:* While conventional FDI implementation environments (i.e. Fortran, Ada, C) support mainly sequential procedural knowledge constructs, expert systems support a hybrid knowledge representation allowing both structural declarative knowledge and sequential procedural knowledge. For instance, the description of the interconnections between the various actuators, surface position sensors and hydraulic lines require a topological knowledge representation capability ideally suited for an expert system implementation.

- *Rapid prototyping with successive refinement:* Expert systems offer an environment where FDI systems can be rapidly prototyped. This is mainly due to the inheritance mechanism supported in hybrid shells and the data-sensitive unordered rules (as opposed to sequential instructions) in the rule based programming paradigm. Moreover, expert systems can be incrementally modified by adding to their knowledge base.

- *Explanation Facility*: Expert systems offer a facility for explaining the reasoning behind the reached decisions. This can be useful not only in advising the pilot (using, for instance, a positive pilot alert system) but also in formalizing the global FDI decision logic.

Expert systems can totally replace some modules in Figure 4.1.1-1. For instance, the hypothesis generator in Figure 4.1.1-1 can be replaced with an expert system with its own knowledge base. Expert Systems can also enhance global FDI performance by augmenting their decision logic in a supervisory role. For instance, the Data Base would include assertions to be used in reasoning by the expert system. Hence, the Data Base would take inputs from the predictor, correlator, detection and isolation test, and reconfiguration logic. For instance, the predictor states can be used to assert events such as "the aircraft is executing a 9g pullup maneuver."

The Rule Base can include the rules describing the inaccurate behavior of the model used in the design of the conventional global FDI algorithm. These rules in a monitoring application may take the form: "If the aircraft is executing a pitchup maneuver greater than 9 g's, and if the FDI algorithm indicates surface damage, then the detection test indication is a false alarm." Ideally, the system knowledge easily expressed numerically should be incorporated into the conventional global design model whereas the system knowledge easily expressed as rules would get incorporated into the expert system Rule Base.

The application of expert systems technology to global FDI systems is not limited to the decision logic portion of these systems. In fact, object oriented hybrid shells are more appropriate in the implementation of FDI systems since they support knowledge constructs (objects, inheritance, methods, topological hierarchy) ideally suited for developing knowledge based systems. In particular, these hybrid shells support the representation of both structural and functional knowledge. Structural knowledge encompasses (physical units such as actuators, hydraulic systems, FCS, surface position sensors, FCS sensors, etc. and interconnection between these physical units such as wires, hydraulic lines, forces, moments, etc). In contrast, functional knowledge encompasses the functional behavior of these physical units captured as methods (procedural code representing input/output behavior, consistency check, etc.).

The major problem in implementing global FDI systems as expert systems is the real time requirements. Current commercially available expert system building tools (shells) are not generally applicable to building expert systems for onboard applications due to the following reasons (Laffey et al. 88): 1) the shells are not fast enough; 2) the shells have insufficient facilities for temporal reasoning; 3) the shells are not easily embeddable into conventional high level programming languages and most cannot run on numeric microprocessors used for embedded applications; 4) the shells have insufficient facilities for devoting attention to significant events; 5) the shells are not designed to accept onboard sensor data; 6) the shells have no integration with a real-time clock and do not handle hardware interrupts; and 7) the shells cannot provide guaranteed response times.

As discussed in (Gupta 85), most interpretive expert system shells spend *90% of their time in matching the current facts against the antecedent of rules in their rule base*. Hence, an expert system development approach where the enterpretive processing is performed off-line would offer a substantial execution time improvement. Similarly, the execution efficiency is a strong function of the knowledge representation facilities employed in the expert system shell. For instance, an approach based on multiple hierarchical representations of a physical system and using forward chaining would have a linear execution time complexity as compared to a rule based system with forward chaining having exponential time complexity.

For ease of integration into conventional high level programs, programming language of the expert system shell is an important choice. For instance, the choice of a programming language commonly used for embedded applications such as Ada or C would be advantageous from an integration viewpoint. Moreover, such an expert system would be easily portable to microprocessors commonly used for embedded applications (e.g., 1750A, 80386, 68020).

Moreover, the language constructs for handling real-time issues (tasking, interrupt servicing, exception handling) would be available to such an expert system development tool.

As discussed by Duke et al. (86), what is needed for real-time onboard expert systems development is a knowledge compiler for converting the developed knowledge base into a conventional program, thus retaining the desirable attributes of the expert system during the development stage while producing an efficient conventional code for a target embedded microprocessor.

One such shell is the System Description Language Processor (SDLP) (Edwards and Caglayan 88) which allows the specification of topological and procedural application knowledge for time-critical applications using a System Description Language (SDL), the *interactive development* of an expert system based on this specification, and the integration of a *compiled version* of this knowledge into a conventional time-critical application. SDLP in an object-oriented shell written in Ada.

4.1.2. Rule-Based Representation of FDI Algorithms

In order to demonstrate the feasibility of implementing FDI algorithms as an expert system, the detection and isolation logic of the CRCA global FDI algorithm is written as a set of production rules. Figure 4.1.2-1 shows the FORTRAN implementation of the current CRCA global FDI detection logic. In this section of code, values associated with each surface are evaluated in order to determine if surface damage has occurred. First, a surface's damage estimate (`PHEI(I)`) is evaluated to ensure that it is within the valid range. Next, the sum of the surface's likelihood ratio (`ALAMDA(I)`) and surface's detection threshold (`TRSHDE(I)`) are compared to the no-fail filter's likelihood ratio (`ALAMDA(NFT1)`). If the no-fail filter's likelihood ratio is larger than the sum, then damage detection flag is set to true (`IFLDET = 1`).

```
      DO 10 I=1,NSUR
        ISOSUR(I) = 0
        TTEMP = ALAMDA(I) + TRSHDE(I)
        IF( (ALAMDA(NFT1).GT.ALAMDA(I)+GRDTA(10)) ) THEN
            IF((PHEI(I).LE.ESTHIG).AND.(PHEI(I).GE.ESTLOW)) THEN
             KNTLR(I) = KNTLR(I) + 1
             KNTFAL = KNTFAL + 1
             INDALM(KNTFAL) = I
             IF(ALAMDA(NFT1).GT.TTEMP) TH"'N
              IFLDET = 1
               KNTLRD(I) = KNTLRD(I) + 1
            END IF
             IF(ALRMIN.GT.ALAMDA(I)) THEN
              ALRMN = ALAMDA(I)
              INDMIN = I
            END IF
          END IF
        END IF
10    CONTINUE
```

Figure 4.1.2-1: Surface Damage Detection FORTRAN Code

A more comprehensible form of this logic can be written using three CLIPS production rules, as shown in Figure 4.1.2-2. The first rule determines if a surface's damage estimate is within the valid range. If it is, the fact ?surface estimate is ok asserted into the fact list. The second rule determines if the sum of the surface's likelihood ratio and the surface's detection threshold is greater than the no-fail filter's likelihood ratio. If it is, the fact ?surface likelihood-ratio-beats-detection-threshold is asserted into the fact list. Finally, damage is detected if both the facts ?surface estimate is ok and ?surface likelihood-ratio-beats-detection-threshold are in the fact list.

```
(defrule determine-if-surface-estimate-is-ok
   (surface-estimate ?surface ?surf-est)
   (surface-estimate-min ?surf-est-min)
   (surface-estimate-max ?surf-est-max)
   (test (< ?surf-est ?surf-est-max))
   (test (> ?surf-est ?surf-est-min))
=>
   (assert (?surface estimate is ok))
)

(defrule determine-if-surface-lr-beats-detection-threshold
    (likelihood-ratio ?surface&~no-fail-filter ?surf-lr)
    (likelihood-ratio no-fail-filter ?nff-lr)
   (detection-threshold ?surface ?surf-dt)
   (test (> ?nff-lr (+ ?surf-lr ?surf-dt)))
 =>
  (assert (?surface likelihood-ratio-beats-detection-threshold))
)

(defrule detect-surface-damage
  (?surface estimate is ok)
    (?surface likelihood-ratio-beats-detection-threshold)
=>
  (assert (surface-damage has been detected))
)
```

Figure 4.1.2-2: Surface Damage Detection CLIPS Code

4.1.3. Integration with Existing FDI Software

Integrating the expert system implementation of the global FDI algorithms with existing FDI software requires embedding the CLIPS rules into the existing FDI FORTRAN routines. Since, CLIPS is based on the C programming language, embedding CLIPS into FORTRAN requires the use of special programs for passing information between CLIPS and FORTRAN. Figure 4.1.3-1 illustrates embedding the CLIPS implementation of the surface damage detection and isolation logic into the existing FORTRAN code.

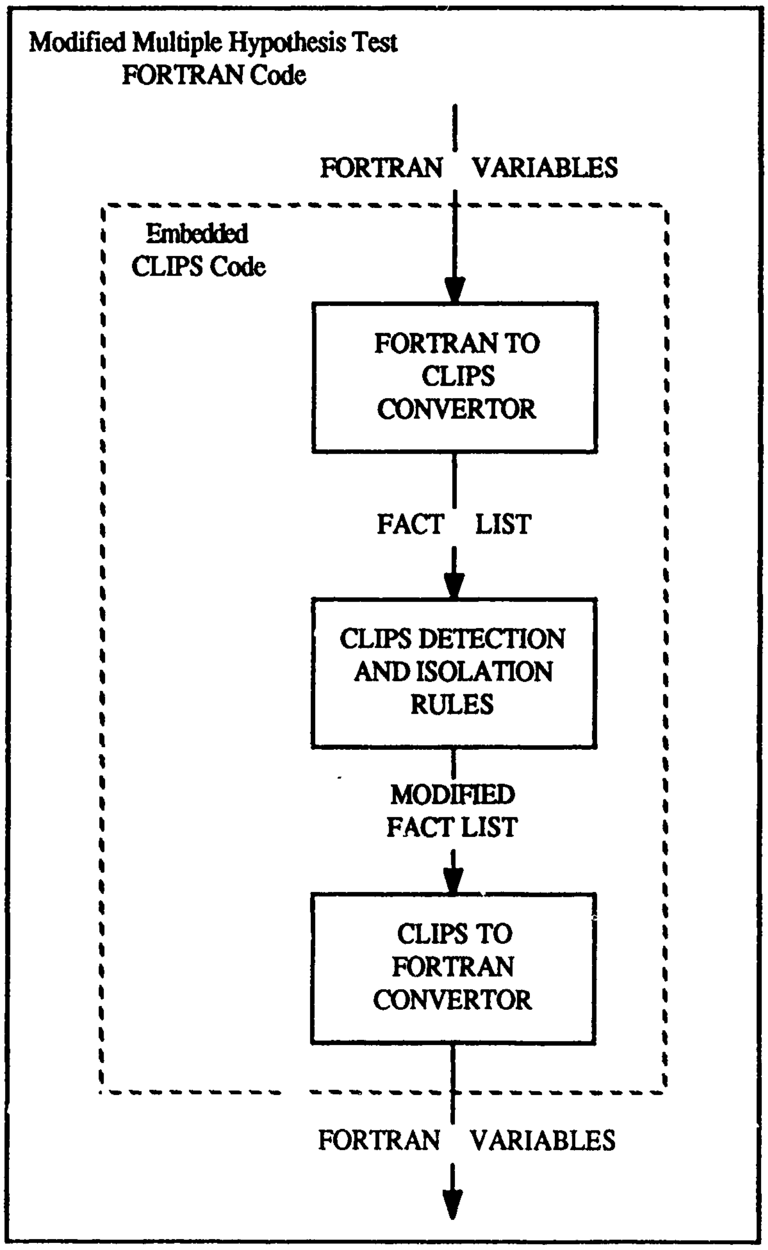
Modified Multiple Hypothesis Test FORTRAN Code
FORTRAN VARIABLES
Embedded CLIPS Code
FORTRAN TO CLIPS CONVERTOR
FACT LIST
CLIPS DETECTION AND ISOLATION RULES
MODIFIED FACT LIST
CLIPS TO FORTRAN CONVERTOR
FORTRAN VARIABLES

Figure 4.1.3-1: CLIPS Embedded In FORTRAN

Accessing the CLIPS rules from inside the FORTRAN code is accomplished by using the FORTRAN *CALL* statement to start the execution of a specialized FORTRAN subroutine. Before the production rules can begin checking for surface damage, the FORTRAN variables needed by the rules must be converted into facts and placed in the fact list. As shown in Figure 4.1.3-1, the inputs to the FORTRAN-to-CLIPS-convertor are FORTRAN variables and the output is the fact list. For example, the surface damage detection rules (Figure 4.1.2-2) need access to the detection thresholds in order to fire. The FORTRAN-to-CLIPS-convertor takes the FORTRAN array variable for detection thresholds, `TRSHDE(I)`, and converts it into the fact (detection-threshold ?surface ?surf-dt). The index, `(I)`, indicates a surface and after conversion is placed into the ?surface slot in the detection-threshold fact. The value for the detection threshold is placed in the ?surf-dt slot. Once the conversion is complete the entire fact is placed into the fact list.

After the fact list has been created, the detection and isolation rules become active and perform the detection and isolation tests. When all rules have finished firing a CLIPS-to-FORTRAN-convertor transforms the modified fact list, which now contains information regarding surface damage, into FORTRAN variables and passes them back to the FORTRAN routine. At this point the FORTRAN routine continues execution.

4.2. Neural Network Implementation of Dynamic Thresholding

As discussed in Chapter 1, there are two different approaches in using neural nets in global FDI design problems. The first one is the direct implementation of the global FDI system as a neural network. The second is to embed neural networks into conventional FDI algorithms. In the first approach, the usual problems associated with an algorithmic approach (e.g. modelling errors, environmental disturbances) would be mapped into the domain of neural network design (e.g. the determination of training data). The second approach is more appropriate for the conventional design stages which do not yield themselves to a well defined algorithmic solution. For instance, the process and measurement noise covariances of a linear design model representing the modelling error uncertainty compared to the underlying nonlinear aircraft model, the no-fail filter estimation error covariance expressions under modelling errors, and likelihood ratio dynamic thresholds compensating the effect of modelling errors are suitable candidates for using a neural net approach.

Here we describe the design of a neural net for determining the likelihood ratio dynamic thresholds in the CRCA global FDI algorithm. Figure 4.2-1 shows the block diagram of the neural net design process. As seen from the figure , the first step is to create a solution (input/output pairs) database. The solution database can be built either by grouping physically related input/output pairs or by grouping all possibly related input/output pairs. For instance, in the second approach, the pilot inputs, actuator commands, surface position sensor measurements, FCS sensor measurements and likelihood ratio outputs can serve as the initial database.

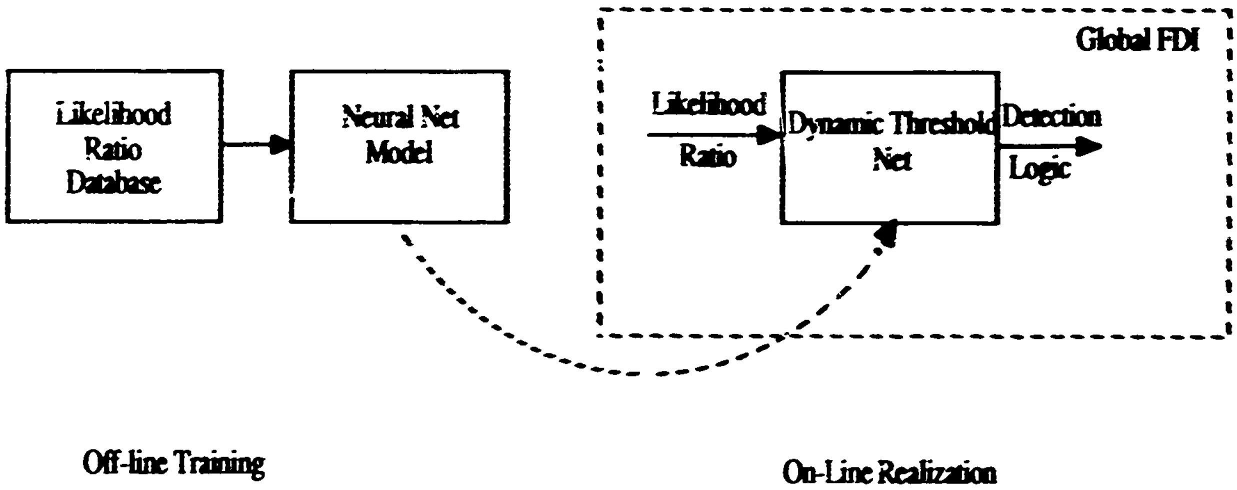

Figure 4.2-1: Neural Net Design Process

The next step in the design is the neural net modelling of the solution database. For a given interconnect topology, the neural net solution can be used to weed variables which do not have an impact on the likelihood ratio outputs (i.e. by observing the links with zero gains). Finally, known functional dependencies can be supplied as inputs for increased representation efficiency.

Figure 4.2-2 shows the two layer neural net used for modelling the dynamic threshold for the left canard likelihood ratio. As seen from the figure, the inputs into the neural net are the commanded and actual roll rates, and the commanded and actual normal accelerations. A ten node intermediate layer with a fully connected topology has been employed. Figure 4.2-3 compares the neural net dynamic threshold with the one in the current CRCA design. In the current CRCA design, the dynamic thresholds have been conservatively scheduled as a function of pilot commands. As seen from Figure 4.2-3, the neural net approach yields a less conservative bound over the likelihood ratio time history. Here, the likelihood ratio excursions are solely due to modelling errors induced by maneuvers shown in Figures 4.2-4 and 4.2-5.

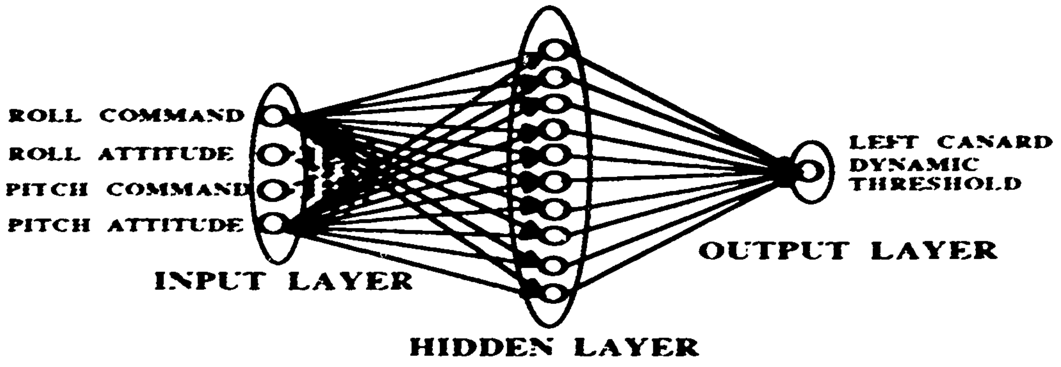

Figure 4.2-2: Two Layer Dynamic Threshold Neural Network

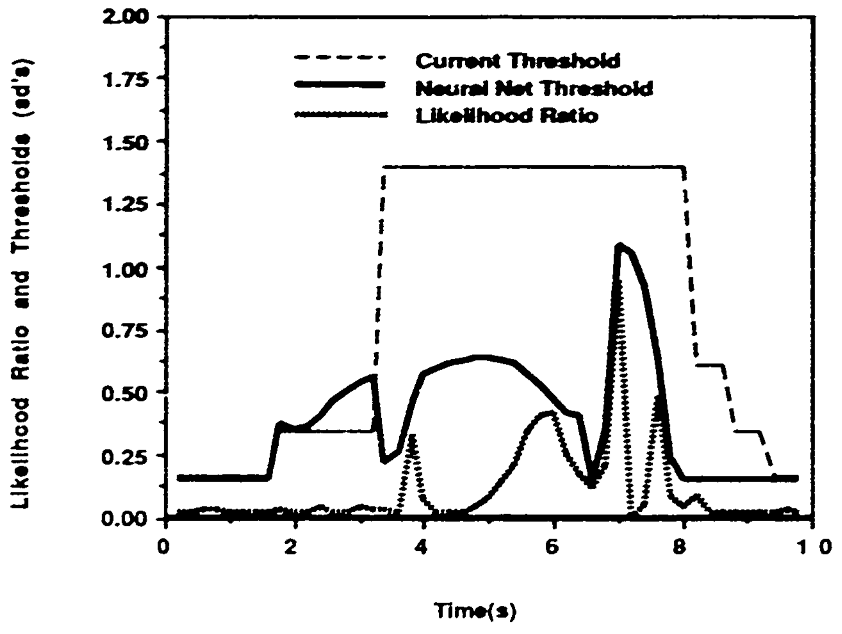

Figure 4.2-3: Neural Net for Dynamic Threshold

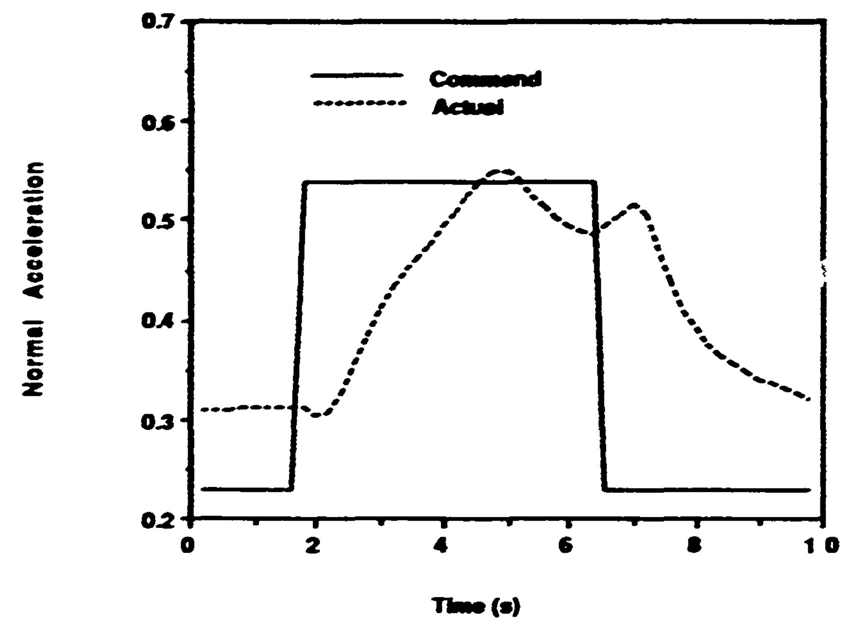

Figure 4.2-4: Pilot g-Command and CRCA Response

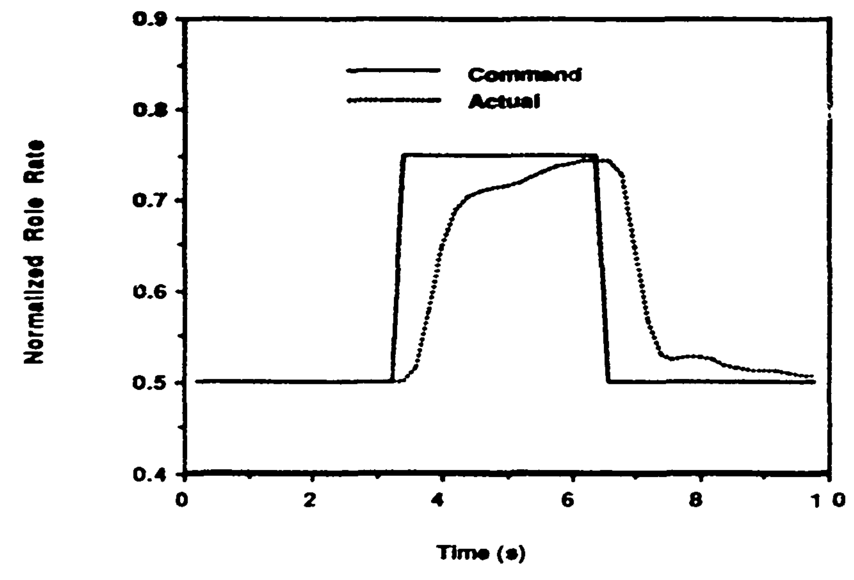

Figure 4.2-5: Pilot p-Command and CRCA Response

5. FULL SCALE RESEARCH PROTOTYPE REQUIREMENTS

Here, we outline the desirable attributes of a full scale research prototype for a Computer Aided Engineering (CAE) solution to the global FDI design problem. The proposed FDI/CAE full scale research prototype consists of a Design Assistant and a Development Environment. Design Assistant assists a designer in optimizing the global FDI design over the entire flight and failure/damage envelopes while the Development Environment supports the implementation of FDI systems as knowledge based expert systems and neural networks.

5.1. Global FDI Design Assistant

A full scale FDI/DA will be an expanded and enhanced version of the FDI/DA prototype described in previous sections of this report. The primary goal of an FDI/DA is to aid the designer in efficiently determining the global FDI design parameters which produce the best FDI performance for the current global FDI strategy. A secondary goal is to help an inexperienced FDI designer learn the design process.

In order to design the global FDI, performance data, generated during simulation runs, must be analyzed. Performing simulation runs and analyzing the data is a time consuming process, so automation of this process is desirable and feasible as demonstrated by the Phase I effort. A full scale research prototype should be able to set up and execute simulation runs as well as analyze performance data beyond the simple discrete event detection (false alarms, misdetection) capability provided in Phase I. Hence, the ultimate objective is to capture the high level information processing capability of an FDI design expert in analyzing the graphic outputs of various performance related variables.

A full scale FDI/DA must have a knowledge base containing all the knowledge necessary for designing the global FDI for optimal performance over the entire flight and damage envelopes. Capturing this knowledge will require formalizing FDI design expert's reasoning during all phases of the design process and transferring their expertise into a structured knowledge base. In particular, the full scale research prototype should include all of the design tradeoffs identified in the FDI Criteria Study (Caglayan, Allen, and Rahnamai 89).

A full scale FDI/DA will contain an explanation facility for explaining to the designer why the suggested actions should be taken and why conclusions were reached.

5.2. Global FDI Development Environment

The global FDI development environment will allow the global FDI designer an efficient environment for developing and evaluating global FDI strategies using artificial intelligence techniques. The global FDI development environment will support the rapid prototyping of FDI strategies which can be successively refined. The designer will be able to experiment with using different programming techniques (knowledge based expert systems, neural networks) in different sections of the global FDI system design to determine which strategies work l ;st for the given section. Using an object-oriented programming development approach, a real time version of the current strategy will be maintained.

The FDI development environment will support the implementation of a global FDI system as a knowledge based expert system supporting reasoning from both structural and functional knowledge. Hence, this expert system development environment will support the specification of topological and procedural FDI knowledge. Furthermore, the development environment will support the interactive development of the knowledge based expert FDI system, and the integration of a compiled version of this knowledge into a real-time version.

The FDI development environment will also support the development of nonalgorithmic subsystems for FDI design using neural networks, and the embedment of the neural net subsystems into the overall FDI design. The development environment will also support the experimentation and evaluation of direct use of neural nets in the FDI design.

6. CONCLUSIONS AND RECOMMENDATIONS

In this study, we have shown the following:

- We have shown the feasibility of capturing FDI design expert's knowledge as a rule based expert system (FDI Design Assistant) by demonstrating such an approach in designing the static thresholds for the current CRCA global FDI design.
- We have investigated the implementation of the global FDI as an expert system, and demonstrated feasibility by implementing the CRCA global FDI detection logic as an expert system, and embedding it into the current global FDI software.
- We have investigated the integration of neural networks into global FDI, and demonstrated feasibility by training a neural net modelling the likelihood ratio dynamic thresholds in the current CRCA global FDI system.

Based on the successful results of our Phase I study, we recommend the development of a full-scale research prototype Computer Aided Engineering (CAE) solution for designing, evaluating and embedding FDI systems. Our proposed FDI/CAE system consists of two major components:

FDI Design Assistant: We recommend the refinement and expansion of the FDI Design Assistant developed under the Phase I effort. The full-scale research prototype would cover all stages of the FDI design process. Moreover, the expansion would enable the coverage of full flight maneuver and damage scenarios. We believe that the rule-based programming paradigm employed in Phase I is sufficient for the proposed full scale Design Assistant development effort. The proposed FDI Design Assistant will be able to set up and execute simulation runs, analyze simulation results, assess FDI performance based on this analysis, and decide on the best course of action for design iteration using its knowledge about the FDI design tradeoffs.

FDI Development Environment: We recommend the construction of an FDI Development Environment enabling the implementation of global FDI using artificial intelligence techniques of expert systems and neural networks. We believe that the rule based programming paradigm employed in Phase I is not sufficient for the proposed Phase II Development Environment. We propose to use an object-oriented expert system development tool for building such an environment so that a model-based approach can be taken in representing structural and functional knowledge. Similarly, the environment would also enable the use of neural nets either directly as damage/failure classifiers or indirectly as nonalgorithmic modelling inaccuracy modellers. Hence, the Development Environment would enable the implementation and evaluation of competing techniques and the incorporation of the best ones into an overall design.

7. REFERENCES

Anderson J. A. and Rosenfeld, E., *Neurocomputing - Foundations of Research,* The MIT Press, Cambridge, MA, 1988.

Anderson, J.R., *The Architecture of Cognition,* Harvard University Press, Cambridge, MA, 1983.

Brown, J.S., Burton, R.R., and Bell, A.G., "SOPHIE: A Sophisticated Instructional Environment for Teaching Electronic Troubleshooting," Bolt, Beranek and Newman, Inc. Report No. 2790, Cambridge, MA 1974.

Buchanan, B.G., and Feigenbaum, E.A., "DENDRAL and meta-DENDRAL: Their Applications Dimension," *Artificial Intelligence,* Vol. II, 1978.

Caglayan, A.K., Allen, S.M., and Rahnamai, K., "Failure Detection, Isolation, and Estimation for Reconfiguration of Aircraft Flight Control Systems Subjected to Actuator Failure and Surface Damage," Charles River Analytics Inc., Cambridge, MA, WRDC-TR-89-3058, June 1989.

Caglayan, A.K., Allen, S.M., and Rahnamai, K., "Failure Detection and Isolation (FDI) Criteria for Control Reconfiurable Combat Aircraft," Technical Report R9002, Contained in AFWAL-TR-88-3118 Vol. IV, *Control Reconfigrurable Combat Aircraft Volume IV CRCA Performance Expansion*, Charles River Analytics Inc., Cambridge, MA, November 1989.

Caglayan, A.K., Allen, S.M., and Wehmueller, K., "Evaluation of a Second Generation Reconfiguration Strategy for Aircraft Subjected to Actuator Failure/Surface Damage," *NAECON 88*, Dayton, OH, 23-27 May 1988.

Caglayan, A.K., Rahnamai, K., Moerder, D.D., and Halyo, N., "A hierarchical Reconfigruation Strategy for Aircraft Subjected to Actuator Failure/Surface Damage," Charles River Analytics Inc., Cambridge, MA, AFWAL-TR-87-3024, May 1987.

Chandler, P.R., "Self-Repairing Flight Control System Reliability and Maintainability Program Plan," AFWAL/FDL, February 1984.

Cohen, M.A. and Grossberg, S., "Absolute Stability of a Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks," *IEEE Transactions on SMC*, Vol. 13, No. 5, October 1983.

Davis, K., "Diagnostic Expert System for the B1B," *IEEE AES Magazine*, April, 1988.

Davis, R., "Diagnostic Reasoning from Structure and Behavior," *Artificial Intelligence*, Vol. 24, pp. 347-410, 1984.

Duda, R.O., et al., "Development of the PROSPECTOR Consultation System for Mineral Exploration Inc.," Final Report, Project No. 6415, SRI International Inc., Menlo Park, CA, 1978.

Duke, E.L., Regenie, V.A., Brazee, M., and Brumbaugh, R.W., "An Engineering Approach to the Use of Expert Systems Technology in Avionics Applications," NASA TM 88263, 1986.

Edwards, S.J. and Caglayan, A.K., "Expert Systems for Real-Time Monitoring and Fault Diagnosis," NASA CR 179441, April 1989.

Fikes, R. and Kohler T., "The Role of Frame Based Representation in Reasoning," *Communications of the ACM*, 1985.

Forbus, K., "Qualitative Physics: Past, Present and Future," *Exploring Artificial Intelligence*, Morgan Kaufman Publishers, 1988.

Forgy, C.L., "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificial Intelligence*, Vol. 19, 1982.

Giarratano, J. C. "CLIPS Reference Manual, V4.2," NASA JSC-22948, April 1988.

Grossberg, S., *Studies of Mind and Brain*, Boston: Reidel, 1982.

Gupta, A., "Parallelism in Production Systems: The Sources and the Expected Speed Up," *Expert Systems and their Applications, Fifth International Workshop*, Avigon, France, 1985

Hecht-Nielsen, R., "Artificial Neural Systems Technology," TRW Rancho Carmel AI Center, San Diego, CA, June 1986.

Laffey, T.J., et al, "Real-Time Knowledge Based Systems," *AI Magazine*, Vol. 9, No. 1, Spring, 1988.

Masui, S., McDermott, J. and Sobel, A., "Decision Making in Time Critical Situations," *Proceedings, IJCAI - 83*, 1983.

Myers, M.H., "Some Speculations on Artificial Neural System Technology," *IEEE Paper,* No. 0547-3578, 1986.

Pisano, A.D. and Jones, H.L., "An Expert Systems Approach to Adaptive Tactical Navigation," *Proceedings of the First IEEE Conference on AI Applications*, December 1984.

Rasmussen, J., "The Role of Hierarchical Knowledge in Decision Making and System Management," *IEEE Transactions on SMC* ,Vol. 15, No. 2, 1985.

"Reconfiguration Strategies for Aircraft with Flight Control Systems Subjected to Actuator Failure/Surface Damage," Lear Siegler Inc., AFWAL-TR-86-3110, March 1987.

"Reconfiguration Strategies for Aircraft with Flight Control Systems Subjected to Actuator Failure/Surface Damage," Scientific Systems Inc., AFWAL-TR-86-3079, December 1986.